SPECIFICATION

Electronic Version 1.2.8

Stylesheet Version 1.0

Method And System For Determining Receipt Of A Delayed Cookie In A Client-Server Architecture

Background of Invention

[0001] 1. *Field of the Invention* The invention pertains to the field of client-server electronic communication, and more particularly to use of a cookie or token to provide and maintain state information.

[0002] 2. *Description of the Related Art* The typical client-server architecture using a client browser and server generated or provided web pages is generally considered a stateless environment. To provide and maintain state information, it is known to use tokens or "cookies".

[0003] A client user logging into a site may receive many different cookies, each with its respective purpose. In some applications, the user receives a cookie that is used to select a particular webserver for load balancing, a cookie representing the user's application session, one or more cookie(s) representing the user's session used for Single Sign On with other systems, and multiple GET ACCESS cookie(s) used for Single Sign On with other GET ACCESS Systems.

[0004] The generation of some cookies, namely the GET ACCESS cookies, can take an extended period of time. In the case of the GET ACCESS cookies, the application server, on behalf of the user performs a simulated login. During the login process the application server connects to the remote system over https, screen scraps the html, logs in as the user, and retrieves the cookies from the resulting http headers. If this login fails a change password is performed on behalf of the user and the login

reattempted. All of this could take quite some time degrading login performance even though the user does not need these cookies immediately upon entering the application.

[0005] What is needed is a system and method to allow a user into a site with only the cookies that are immediately necessary. Other cookies, such as the GET ACCESS cookies, are delivered asynchronously as they become available.

[0006] The preceding description is not to be construed as an admission that any of the description is prior art to the present invention.

Summary of Invention

[0007] In one aspect, the invention provides a method and system for sending information to a client browser comprising sending a request from a client to a server, determining whether the request includes a token, and if the token is not included in the request, determining whether the token is available to send to the client, wherein the token was created responsive to an earlier request from the client to the server.

[0008] In one aspect, the invention further comprises if the token is included in the request, making a record that the client has the token.

[0009] In one aspect, the invention further comprises if the token is available to send to the client ,sending a response to the client with the token.

[0010] In one aspect, the invention further comprises if the token is not available to send to the client, sending a response to the client without the token.

[0011] In one aspect, the invention further comprises determining if a timer exceeds a predetermined value, and if the timer does not exceed the predetermined value, determining whether the token is available to send to the client, and if the timer does not exceed the predetermined value, sending a response to the client without the token.

[0012]

The specific aspects and advantages of the invention described and illustrated herein are illustrative of those which can be achieved by the present invention and are not intended to be exhaustive or limiting of the possible advantages that can be

realized. Thus, the aspects and advantages of this invention will be apparent from the description herein or can be learned from practicing the invention, both as embodied herein or as modified in view of any variations which may be apparent to those skilled in the art. Accordingly, the present invention resides in the novel parts, constructions, arrangements, combinations and improvements herein shown and described.

Brief Description of Drawings

[0013] The foregoing features and other aspects of the invention are explained in the following description taken in conjunction with the accompanying figures wherein:

[0014] FIG. 1 illustrates a system according to one embodiment of the invention;

[0015] FIG. 2 illustrates steps in a method according to one embodiment of the invention; and

[0016] FIG. 3 illustrates steps in a method according to one embodiment of the invention.

[0017] It is understood that the drawings are for illustration only and are not limiting.

Detailed Description of the Drawings

[0018] In one embodiment, a request is made from the client to retrieve an http(s) page from the server (e.g. login form submitted and client expects resulting html). The server then spawns a long running process to retrieve the slow cookies and returns the fast cookies along with the html to the user immediately. The html returned from the server to the client contains a link to a clear gif retrieved over http(s). The clear gif is an image file that contains only transparent data, so appears to be "clear." The clear gif is not a file but rather the output of a program that listens for notifications that the slow cookies have been retrieved. If no such notifications arrive after some period of time (e.g., 20 seconds) the program does not return a clear gif but rather redirects the client to another URL resulting in the same program being run. This is so that the "persistent connection" between the client and the server is not recognized as such and dropped by intermediate proxies that do not allow such connections. Eventually the slow cookies will be retrieved, the clear gif program notified, and the cookies returned in the http headers of the clear gif along with the clear gif itself. After some configurable timeout the clear gif could be returned even if the slow cookies never

arrive to prevent the wasting of resources.

[0019] Referring now to FIG. 1, system 100 according to one embodiment of the invention includes a client 102, a server 104 and a cookie or token generator 106. Client 102 and server 104 are electrically connected to each other by network 108. Server 104 and cookie generator 106 are electrically connected to each other by either network 108, or by network 110. Network 108 can be any of a number of types of wired and wireless networks, such as a local area network (LAN) or wide area network (WAN). In one embodiment, network 108 is the Internet. Network 110 can also be any of a number of types of wired and wireless networks, and may also be the Internet. Depending on the needs of the system, network 108 and 110 may be parts of the same network, or they may be isolated from each other.

[0020] Although not specifically illustrated in FIG. 1, client 102, server 104 and cookie generator 106 each include removable and fixed software code storage media, a processor to run the software code, a memory, input and output devices and network interface devices, all interconnected by a system bus.

[0021] Although in FIG. 1 server 104 and cookie generator 106 are illustrated as separate entities, it is also possible that they are both elements of the same piece of hardware, but can perform different functions.

[0022] Referring now to FIG. 2, system 100 begins at step 202 when client 102 sends a resource request to server 104. This is typically an http or https type of resource request that includes a URL corresponding to the requested resource.

[0023] At step 204, server 104 receives the resource request. Although not illustrated in FIG. 2, at step 204, server 104 also determines whether all the cookies or tokens that should be returned to client 102 exist or are immediately available. If all of the cookies or tokens exist, then the requested resource is returned to the client along with the cookies or tokens. These cookies or tokens that are immediately available are the "fast" cookies.

[0024] Assuming that there are some cookies or tokens that are not immediately available (i.e., they are "slow" cookies), then at step 206, server 104 sends a request to cookie generator 106 to generate a cookie (e.g., the "slow" cookies).

[0025] Without waiting for cookie generator 106 to generate and return the requested cookie, at step 208 server 104 begins to generate or retrieve the html for the requested resource. In addition to fast cookies, which are immediately available, the html includes a link to the clear gif, as a substitute or place-holder for the slow cookie (s).

[0026] Once server 104 has generated or retrieved the html, then at step 210, server 104 sends the html, including the link to the clear gif, to client 102.

[0027] At step 212, client 102 receives the html, including the "fast" cookies and the link to the clear gif.

[0028] At step 214, client 102 renders the html on the browser. Once step 214 is complete, the resource that was requested at step 202, including the associated "fast" cookies has been sent to client 102 from server 104 with the exception of the "slow" cookie(s).

[0029] At step 216, client 102 establishes a connection or sends a request for the clear gif that is represented by the link.

[0030] At step 218, server 104 receives the connection request for the clear gif.

[0031] At step 220, server 104 starts a timer. This timer is used to avoid time-out problems on the browser and also avoid errors from what appears to be a persistent connection between the client and the browser. In one embodiment, the timer is a count-down timer and is set to 20 seconds.

[0032] At step 221, server 104 checks to see if the cookie, requested at step 206, is present in the send queue. Therefore, it is helpful to understand the cookie generation steps that occur after step 206.

[0033] At step 230, cookie generator 106 receives the request to generate a cookie that server 104 sent at step 206.

[0034] At step 232, cookie generator 106 generates the requested cookie. As an example, this could be a GET ACCESS cookie.

[0035] At step 234, cookie generator 106 sends the requested cookie to server 104.

[0036] At step 236, server 104 receives the requested cookie and puts the cookie in a send queue.

[0037] The time required for cookie generator 106 to complete steps 230 through 234 is normally greater than the time required for server 104 to complete steps 206 through 210. For this reason, server 104 normally receives the cookie at step 236 after completing step 210. Although not illustrated, if the requested cookie is received at step 236 before server 104 completes step 210, the requested cookie can be included with the html at step 210. However, it is also possible that even though the cookie is received before server 104 completes step 210, the requested cookie is not included with the html at step 210.

[0038] After starting the timer at step 220, at step 221, server 104 determines whether the requested cookie is present in the send queue. If the cookie is present, then at step 226, server 104 sends the cookie to the client browser as part of the header in the clear gif.

[0039] At step 228, client 102 receives the clear gif, including the cookie in the header, and the browser "renders" the clear gif.

[0040] If, at step 221, server 104 determines that the requested cookie is not present in the send queue, than at step 222, server 104 determines whether the timer, started at step 220, has expired. If the timer has not expired, server 104 loops to step 221 and continues to monitor or check for the cookie in the send queue.

[0041] If at step 222, server 104 determines that the timer has expired, then at step 224, client 102 is provided instructions, such as through a redirect, to reload the process, and loops to step 216.

[0042] In one embodiment, the system and method includes techniques to ensure that the "slow" cookie is received by the client. Referring to FIG. 3, an example of these techniques begins at step 302, with client 102 sending a connection request to server 104, the request including the "clear gif" link. This connection request can be the same connection request illustrated at step 216, or it can be another connection request.

[0043] At step 304, server 104 receives the connection request to the "clear gif" link.

[0044] At step 306, server 104 checks the connection request to see if it includes the "slow" GET ACCESS cookie.

[0045] If at step 308, server 104 determines that the "slow" cookie is present, then at step 310, server 104 makes an entry or record in a buffer or database to indicate that the "slow" GET ACCESS cookie was received by client 102.

[0046] After server 104 makes a record that the "slow" cookie was received, then at steps 324 - 328, server 104 prepares and sends the clear gif without the "slow" cookie. This is because the "slow" cookie has already been received by client 102, as indicated at steps 306 and 308.

[0047] If, at step 308, server 104 determines that the cookie is not present in the request, then at step 312, server 104 checks for see if the requested "slow" cookie has been prepared or is in the queue awaiting transmission to client 102. This can be the same or similar to step 221 in FIG. 2

[0048] If, at step 314, server 104 determines that the "slow" cookie is available for transmission to client 102, then at step 316 server 104 processes the "slow" cookie with the requested clear gif, and at step 318, server 104 sends the clear gif with the "slow" cookie to client 102. At step 320, client 102 receives the clear gif with the "slow" cookie. The slow cookie is held by client 102 and serves to authenticate or authorize subsequent resource requests by client user.

[0049] If, at step 314, server 104 determines that the requested "slow" cookie was not available, then at step 322, server 104 determines whether the timer has expired. This timer can be the same timer illustrated in FIG. 2, or it may be a second timer, started as some point after server 104 receives the connection request sent by client 102 at step 302.

[0050] If at step 322, if server 104 determines that the timer has not expired, it loops to step 312 to continue the check for the "slow" cookie.

[0051] If at step 322, server 104 determines that the timer has expired, then at step 324, server 104 prepares the clear gif without the requested "slow" cookie, and at step 326

sends the clear gif to client 102. At step 328, client 102 receives the clear gif without the requested cookie.

[0052] The techniques that have been described can be implemented in a number of different ways. In one embodiment, the link to the clear gif is manually inserted into the html during page development. In another embodiment, the link to the clear gif is automatically inserted by a plug-in that edits the html output of the application server. In another embodiment, a spacer gif is automatically used, resulting in no changes to the html. This last embodiment is particularly useful where the web site is designed using a "what you see is what you get" ("wysiwyg") editor.

[0053] Although illustrative embodiments have been described herein in detail, it should be noted and will be appreciated by those skilled in the art that numerous variations may be made within the scope of this invention without departing from the principle of this invention and without sacrificing its chief advantages. Such variations include: the clear gif can be reloaded not by http redirects but rather by JavaScript that switches the image; the cookies can be set by a webserver plug-in or html included in every page that checks each time a page is requested to see if the slow cookies have been retrieved; or the cookies can be set by a bounce page when the user clicks a link requiring the cookies. There are various advantages and disadvantages of these variations.

[0054] Unless otherwise specifically stated, the terms and expressions have been used herein as terms of description and not terms of limitation. There is no intention to use the terms or expressions to exclude any equivalents of features shown and described or portions thereof and this invention should be defined in accordance with the claims that follow.